GHASP: A Galileo High Accuracy Service Parser

User Manual

Borio D., Gioia C., Susi M.
February 2023

Table of Contents

Introduction

The Galileo High Accuracy Service (HAS) was declared operational on 24th January 2023, during the annual European Space Conference. Through HAS, Galileo is broadcasting orbit, clock and measurement corrections enabling decimeter level positioning.

HAS corrections are distributed through the E6B signal, which adopts a high-parity vertical Reed-Solomon encoding scheme (Fernández-Hernández et al. 2020). Thus, E6B signals need to be decoded in order to recover the HAS corrections enabling Precise Point Positioning (PPP).

In order to foster the use of HAS corrections, a HAS parser has been developed. This is the user manual of such parser, which has been denoted as Galileo HAS Parser (GHASP).

GHASP supports several data types from different receiver types and converts E6B data messages, recorded as binary streams, into actual PPP corrections. The outputs of the parser are four Comma-Separated Values (CSV) files containing the different correction types. The corrections can be easily loaded using any scientific programming language and used for different analyses in addition to PPP applications.

This manual describes the download and installation instructions. The use of a Graphical User Interface (GUI) is also illustrated in order to simplify the different operations. The output files and their format are detailed. Finally, some plotting functions are illustrated.

Download and Installation Instructions

The GHASP library can be downloaded from the GitHub repository https://github.com/borioda/HAS-decoding. Once the different dependencies are satisfied, one should be able to use the parser directly.

The parser can be operated by either running the main script of the library (*process_cnav.py*) or by using the GUI provided as a Jupyter notebook. Detailed instructions on the use of the GUI can be found in the next section.

The GHASP library relies on a few libraries available for most Python distributions. The following dependencies are needed to run the script version of GHASP without Jupyter notebook GUI:

abc – module providing the infrastructure for defining abstract base classes (https://docs.python.org/3/library/abc.html)

galois – a performant NumPy extension for Galois fields (https://pypi.org/project/galois/). It is used for the operations needed for inverting the Reed-Solomon encoding scheme.

NumPy – fundamental package for scientific computing with Python (https://numpy.org/)

pandas – popular library for data analysis and manipulation (https://pandas.pydata.org/)

struct – library to interpret bytes as packed binary data (https://docs.python.org/3/library/struct.html)

tqdm - efficient implementation of a progress bar (https://tqdm.github.io/)

In order to run the GHASP GUI implemented as a Jupyter notebook the *ipywidgets* library is also needed (https://ipywidgets.readthedocs.io).

GUI Usage

The GHASP library can be easily used through a GUI provided as a Jupyter notebook (https://jupyter.org/). The Jupyter notebook named *Galileo_HAS_Parser.ipynb* can be run through the Jupyter lab/notebook interfaces provided by the main Python distributions. For instance, Jupyter lab and notebook servers are readily available in *Anaconda* (https://www.anaconda.com/) and *WinPython* (https://winpython.github.io/).

A screenshot of the Jupyter notebook developed for facilitating the use of GHASP is shown in Figure 1.

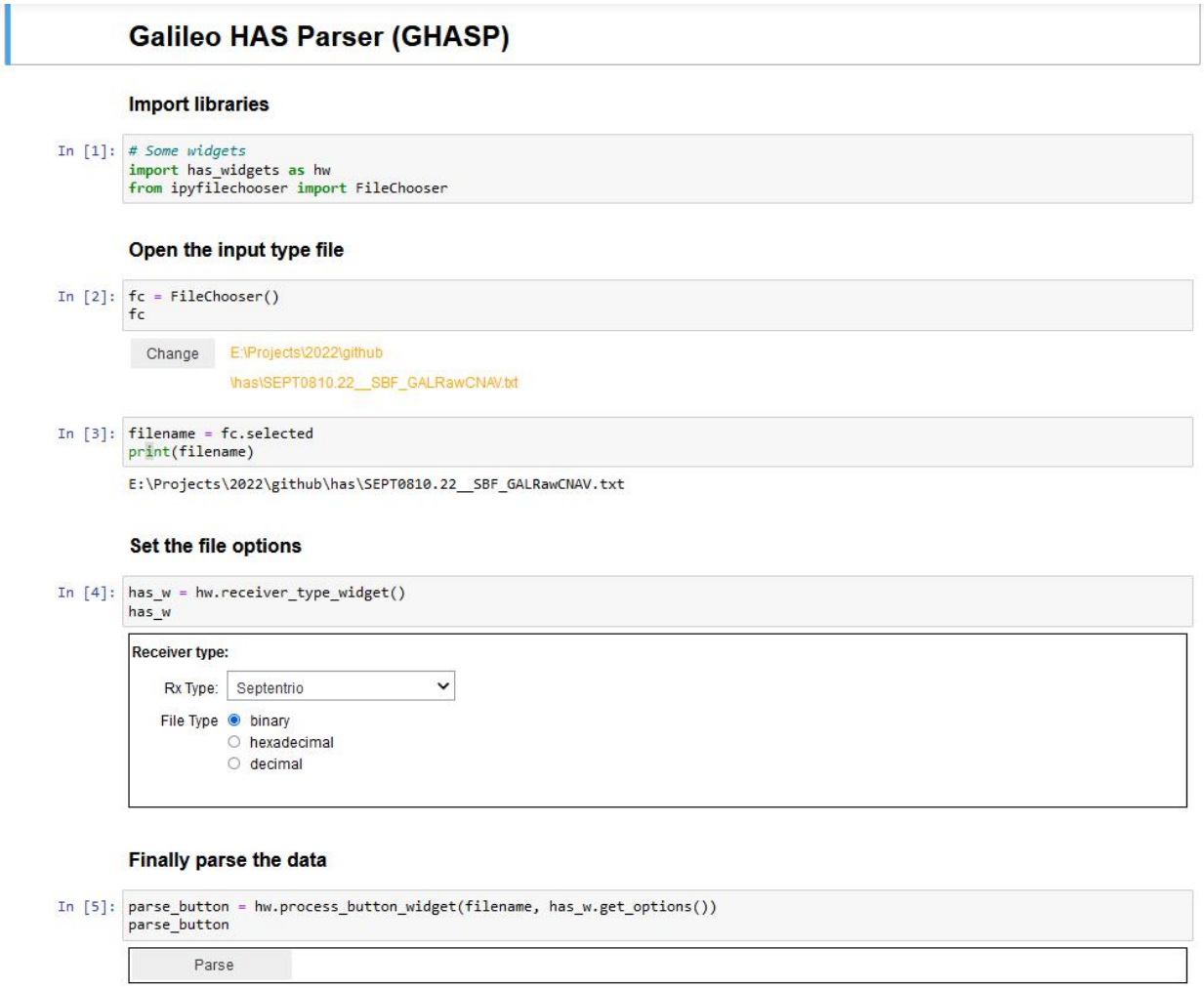

Figure 1: Screenshot of the Jupyter notebook developed to simplify the user interaction with the HAS decoder.

The first box of the notebook is used to load the different libraries required by the GUI. The *has_widgets* library is a wrapper, which packages the functionalities of GHAPS into simple graphical widgets that are then displayed in the notebook. The widgets hide the complexity of the decoder, which can be used without requiring advanced programming skills.

The second element loaded is the *FileChooser* widget from the ipyfilechooser library, which can be easily installed following the on-line instructions available here: https://pypi.org/project/ipyfilechooser/.

The second box in the notebook instantiates a *FileChooser* widget, which is used to select an input file or a directory. If a directory is selected, the parser will try to process all the files in the directory assuming that all the files are of the same type, i.e., generated by the same receiver.

The next box in the notebook simply checks the correctness of the file/directory selected using the *FileChooser* widget. Indeed, it simply prints the path of the file/directory chosen.

Box number 4 create a *receiver_type_widget*, which allows the user to specify the type of input file. Currently, the following files produced by the following receiver types are supported:

Septentrio
Javad
NovAtel

When the Septentrio option is selected, it is possible to further specify the file format. Indeed, Septentrio receivers are provided with software tools allowing one to convert the native Septentrio Binary Files (SBFs) into text files. By selecting the option "binary," the user specifies that the input file is from a Septentrio receiver and in SBF format. If the "hexadecimal" option is selected, it means that the SBF file has been parsed and that the GALRawCNAV message has been extracted and saved into a text file. Thus, the file specified as input for the decoder is a text containing only GALRawCNAV messages. Moreover, the original E6B messages are encoded as hexadecimal characters. The option "decimal" specifies a file format similar to that expected for the "hexadecimal" case. Also in this case, GALRawCNAV messages have been extracted from the SBF file and saved as text. The only difference is that, in this case, the E6B messages are encoded as a sequence of 32-bit decimal values.

Once the input file format has been specified, parsing can start. This is done through the execution of the last box in the notebook, which allocates a single button labeled as "Parse." By pressing this button, the user triggers the parsing of the input files.

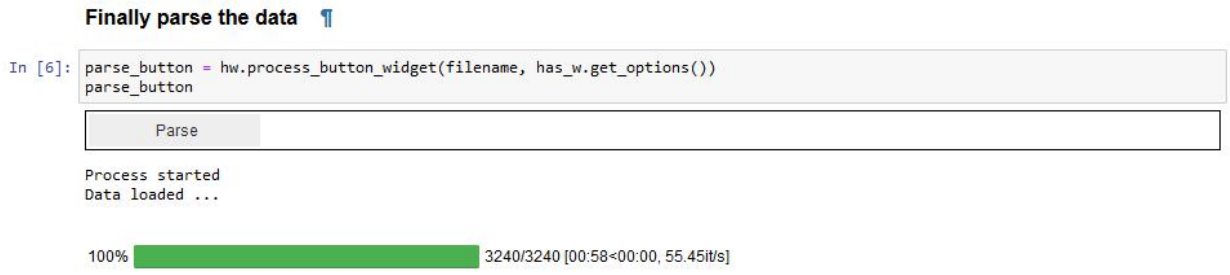

Figure 2: Screenshot of the Jupyter notebook GUI: the user is notified about the advancement of the processing by a progress bar.

The user is notified about the advancement of the processing by a progress bar, which is implemented through the Python library *tqdm* (https://tqdm.github.io/).

As already mentioned, if a directory is selected, the parser will try to process all the files present in the folder. In this case, several progress bars will be sequentially shown.

Output Files

For each input file, a set of four CSV files is produced by the parser. Each output file will inherit its name from the input file plus a suffix indicating the type of corrections it contains. For instance, assume that the input file is named

filename.in

The following output files are produced:

filename_ has_orb.csv: for the orbit corrections
filename_ has_clk.csv: for the clock corrections
filename_ has_cb.csv: for the code biases
filename_ has_cp.csv: for the carrier phase biases

Each file has its structure as specified in the respective file headers. The four cases are discussed in the following.

HAS Orbit Correction Files

The following columns are present in the HAS orbit correction output file:

ToW: the Time of Week in seconds as provided by the receiver. This is not part of the HAS message, but it is extracted by the receiver when processing other GNSS signals. It is used together with the Time of Hour (ToH) to determine the absolute time of applicability of the HAS corrections (see section 7.7 of (European Union, 2022).

WN: the GPS week number

ToH: Time of Hour in seconds. Number between 0-3599 that determines the number of seconds of the time of applicability from the start of the last hour as specified by the Galileo System Time (GST).

IOD: as indicated in the Galileo HAS ICD (European Union, 2022, section 5.1.1.2), this is “a counter that identifies the set of Reference IODs for all corrected satellites of the mask”.

gnssIOD: counter corresponding to the IOD of the orbit and clock data corrected. This data is broadcast by the different GNSSs and identified by an IOD reported here.

validity: validity in seconds of the correction. A correction should be applied only to measurements belonging to the time interval defined by the time of applicability (computed from the ToW and ToH) and the time of applicability plus the validity interval.

gnssID: GNSS ID identifying the GNSS to which a specific correction refers to (European Union, 2022, section 5.2.1.1). Currently only GPS (gnssID = 0) and Galileo (gnssID = 2) are supported.

PRN: number of the satellite to which the correction refers.

delta_radial: the orbit radial correction in meters.

delta_in_track: the orbit in-track correction in meters.

delta_cross_track: the delta cross-track correction in meters.

HAS Clock Correction Files

The columns present in the output file with the HAS clock corrections are described below. The first eight elements in the file are the same as those found in the orbit correction file. They are repeated for completeness.

ToW: the Time of Week in seconds as provided by the receiver. This is not part of the HAS meassage, but it is extracted by the receiver when processing other GNSS signals. It is used together with the Time of Hour (ToH) to determine the absolute time of applicability of the HAS corrections (see section 7.7 of (European Union, 2022).

WN: the GPS week number

ToH: Time of Hour in seconds. Number between 0-3599 that determines the number of seconds of the time of applicability from the start of the last hour as specified by the Galileo System Time (GST).

IOD: as indicated in the Galileo HAS ICD (European Union, 2022, section 5.1.1.2) this is a “a counter that identifies the set of Reference IODs for all corrected satellites of the mask”.

gnssIOD: counter corresponding to the IOD of the orbit and clock data corrected. This data is broadcast by the different GNSSs and identified by an IOD reported here.

validity: validity in seconds of the correction. A correction should be applied only to measurements belonging to the time interval defined by the time of applicability (computed from the ToW and ToH) and the time of applicability plus the validity interval.

gnssID: GNSS ID identifying the GNSS to which a specific correction refers to (European Union, 2022, section 5.2.1.1). Currently only GPS (gnssID = 0) and Galileo (gnssID = 2) are supported.

PRN: number of the satellite to which the correction refers.

multiplier: number between 1 and 4, which multiplies the clock correction.

delta_clock_c0: clock correction in meters. The final clock correction is obtained as the product of the "multiplier" and "delta_clock_c0"

status: status flag derived from the binary value of "delta_clock_c0" (European Union 2022, Table 31). When status is set to 1, the satellite correction shall not be used. For status equal to 2, data are not available. A status value equal to 0 indicates the availability of the correction.

HAS Clock Bias Files

The columns present in the output file with the HAS code biases are described below. The first eight elements in the file are the same as those found in the orbit correction file. They are repeated for completeness.

ToW: the Time of Week in seconds as provided by the receiver. This is not part of the HAS message, but it is extracted by the receiver when processing other GNSS signals. It is used together with the Time of Hour (ToH) to determine the absolute time of applicability of the HAS corrections (see section 7.7 of (European Union, 2022).

WN: the GPS week number

ToH: Time of Hour in seconds. Number between 0-3599 that determines the number of seconds of the time of applicability from the start of the last hour as specified by the Galileo System Time (GST).

IOD: as indicated in the Galileo HAS ICD (European Union, 2022, section 5.1.1.2) this is a "a counter that identifies the set of Reference IODs for all corrected satellites of the mask".

gnssIOD: counter corresponding to the IOD of the orbit and clock data corrected. This data is broadcast by the different GNSSs and identified by an IOD reported here.

validity: validity in seconds of the correction. A correction should be applied only to measurements belonging to the time interval defined by the time of applicability (computed from the ToW and ToH) and the time of applicability plus the validity interval.

gnssID: GNSS ID identifying the GNSS to which a specific correction refers to (European Union, 2022, section 5.2.1.1). Currently only GPS (gnssID = 0) and Galileo (gnssID = 2) are supported.

PRN: number of the satellite to which the correction refers.

signal: code indicating the signal type of the correction. The different signal types can be found in Table 20 of the HAS ICD (European Union, 2022).

code_bias: code bias in meters.

av_flag: availability flag derived from the "code_bias" value as specified in Table 37 of the Galileo HAS ICD (European Union, 2022). An availability flag equal to 0 indicates that the code bias is not available.

HAS Carrier Phase Bias Files

The columns present in the output file with the HAS carrier biases are described below. The first eight elements in the file are the same as those found in the orbit correction file. They are repeated for completeness.

ToW: the Time of Week in seconds as provided by the receiver. This is not part of the HAS message, but it is extracted by the receiver when processing other GNSS signals. It is used together with the Time of Hour (ToH) to determine the absolute time of applicability of the HAS corrections (see section 7.7 of (European Union, 2022).

WN: the GPS week number

ToH: Time of Hour in seconds. Number between 0-3599 that determines the number of seconds of the time of applicability from the start of the last hour as specified by the Galileo System Time (GST).

IOD: as indicated in the Galileo HAS ICD (European Union, 2022, section 5.1.1.2) this is a "a counter that identifies the set of Reference IODs for all corrected satellites of the mask".

gnssIOD: counter corresponding to the IOD of the orbit and clock data corrected. This data is broadcast by the different GNSSs and identified by an IOD reported here.

validity: validity in seconds of the correction. A correction should be applied only to measurements belonging to the time interval defined by the time of applicability (computed from the ToW and ToH) and the time of applicability plus the validity interval.

gnssID: GNSS ID identifying the GNSS to which a specific correction refers to (European Union, 2022, section 5.2.1.1). Currently, only GPS (gnssID = 0) and Galileo (gnssID = 2) are supported.

PRN: number of the satellite to which the correction refers.

signal: code indicating the signal type of the correction. The different signal types can be found in Table 20 of the HAS ICD (European Union, 2022).

phase_bias: phase bias in cycles.

av_flag: availability flag derived from the "phase_bias" value as specified in Table 40 of the Galileo HAS ICD (European Union, 2022). An availability flag equal to 0 indicates that the code bias is not available.

phase_discontinuity_ind: phase discontinuity indicator defined in section 5.2.6.1. This is a counter used to indicate a phase discontinuity in the signal phase. The process of ambiguity fixing should be reinitialized every time this counter is incremented.

Plotting the Results

GHASP is provided with some basic routines that allow one to plot the corrections extracted from the different receiver files.

Four Python scripts are provided as part of the GHASP library:

plot_orb.py: to plot the orbit corrections. The three components of the orbit corrections are displayed in a different subplot. A different figure is generated for each GNSS for which corrections are available. An example of plot generated by the plot_orb.py routine is provided in Figure 3.

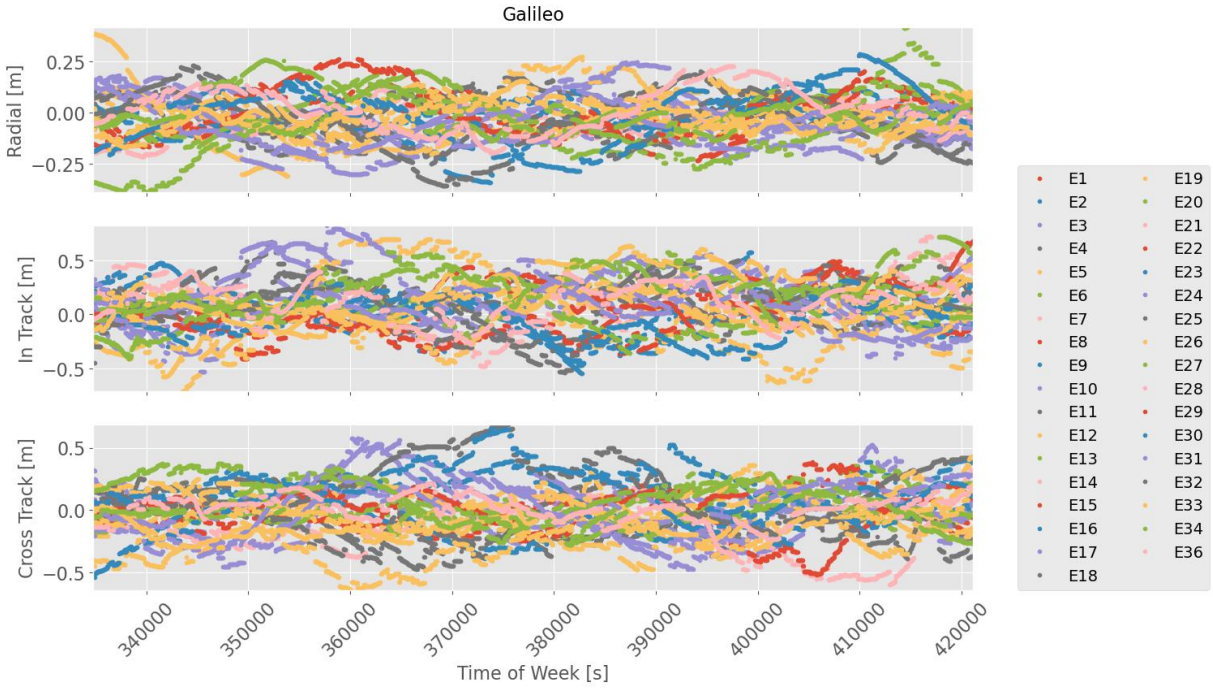

Figure 3: Example of plot with the orbit corrections generated for the Galileo satellites using the plot_orb.py script.

plot_clk.py: to plot the clock corrections. A different figure is generated for each GNSS for which corrections are available. An example of plot generated by the plot_clk.py routine is provided in Figure 4.

plot_cb: to plot the code bias corrections. A different figure is generated for each GNSS for which corrections are available. Different subplots are used to display the corrections for the different signals corrected. An example of plot generated by the plot_clk.py routine is provided in Figure 5.

plot_cp: to plot the phase bias corrections. A different figure is generated for each GNSS for which corrections are available. Different subplots are used to display the corrections for the different signals corrected. An example of plot generated by the plot_clk.py routine is provided in Figure 6.

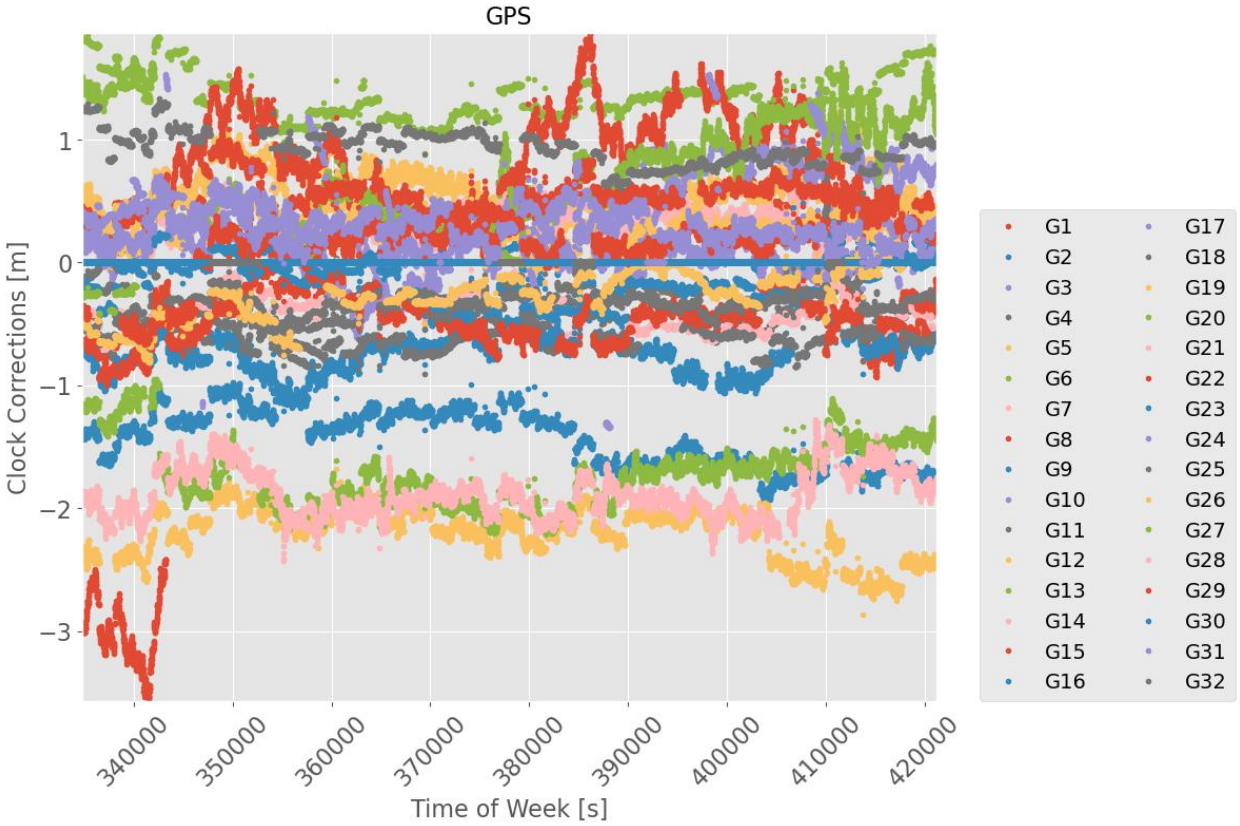

Figure 4: Example of plot with the clock corrections generated for the GPS satellites using the plot_clk.py script.

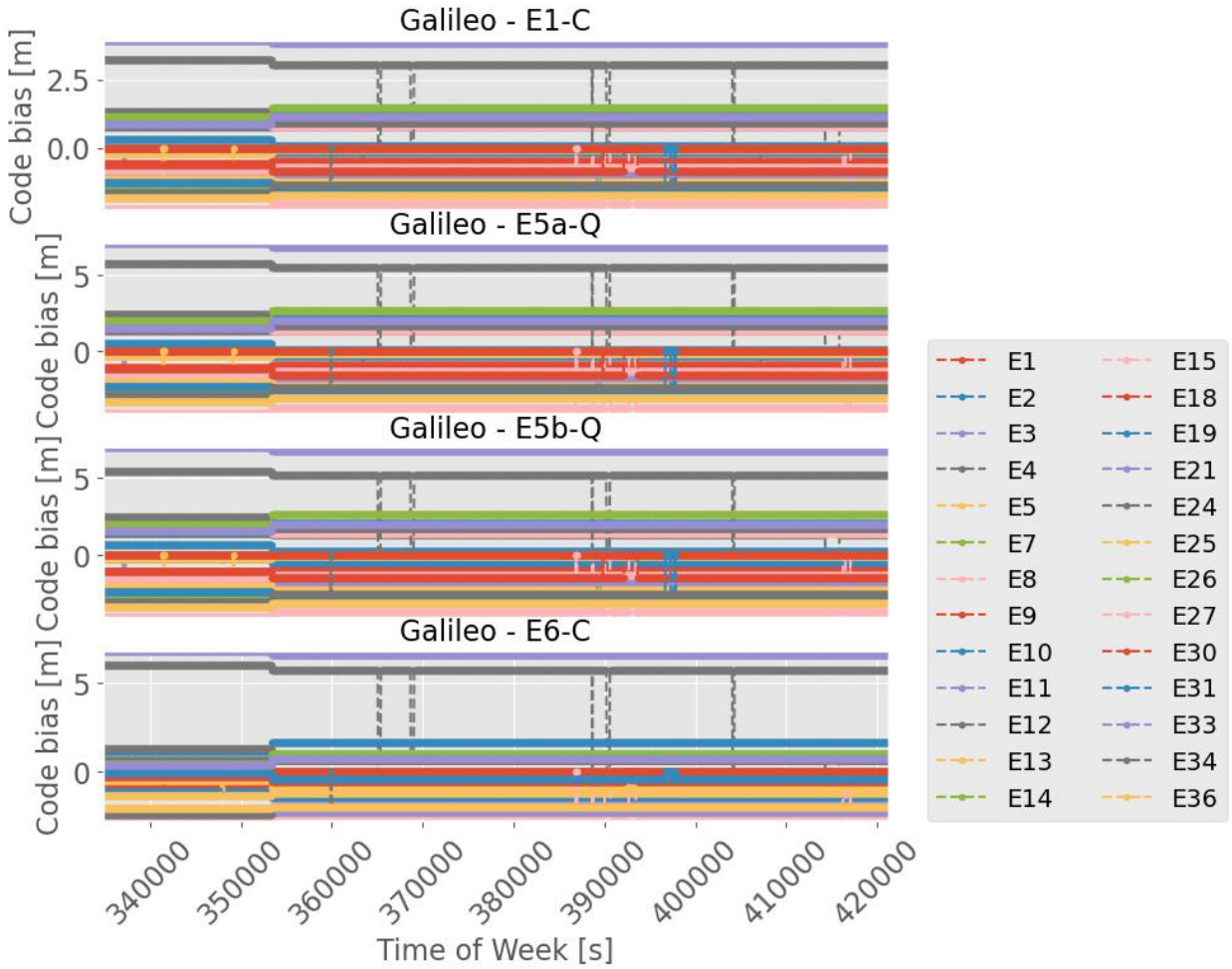

Figure 5: Example of plot with the code bias corrections generated for the Galileo satellites using the plot_cb.py script.

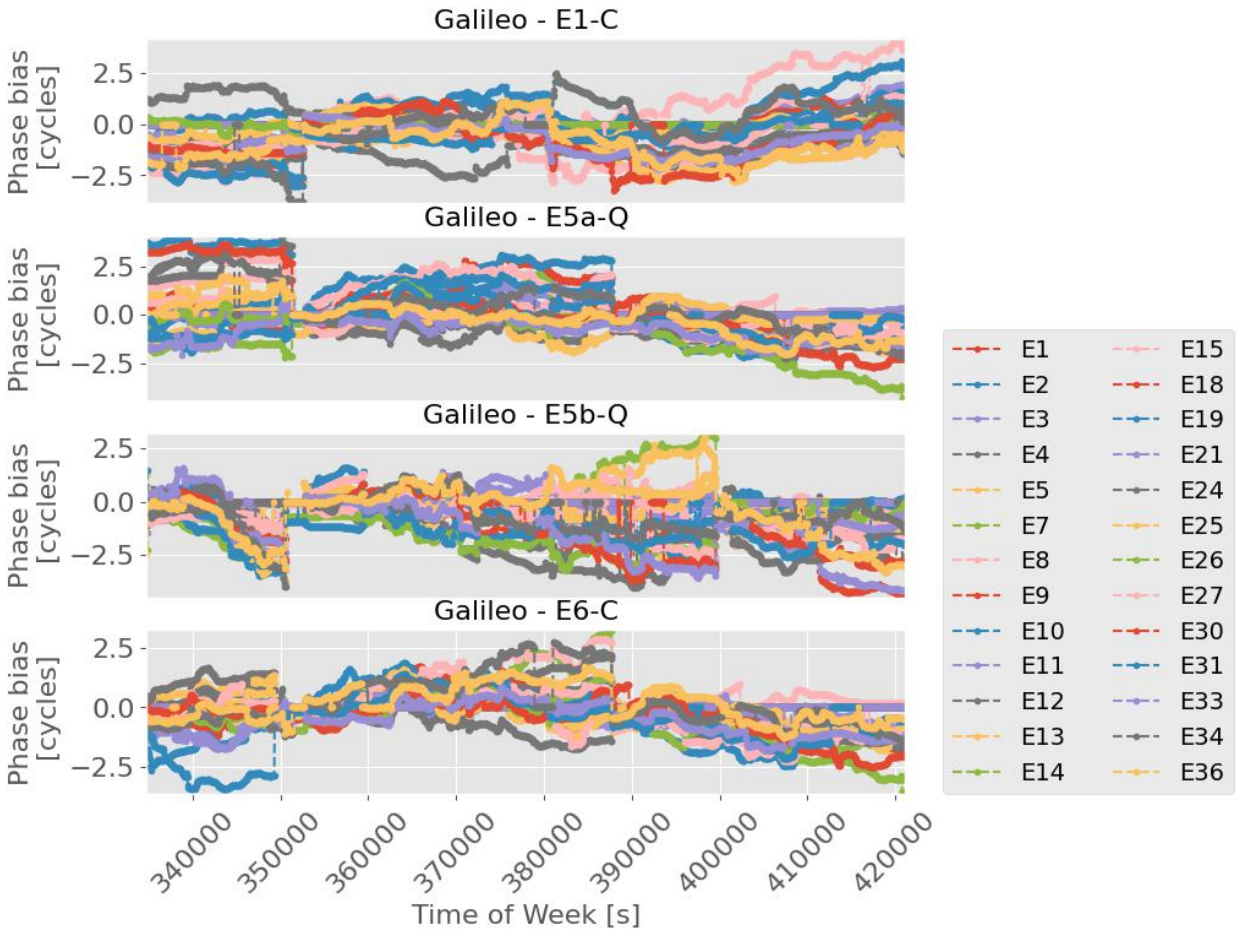

Figure 6: Example of plot with the phase bias corrections generated for the Galileo satellites using the plot_cp.py script.

In addition to the four plotting routines provided as part of GHASP, an additional script has been included (*clk_adev.py*) to experiment with the evaluation of the Allan Deviation (AVED) of the HAS clock corrections. The script relies on the *allantools* library (https://pypi.org/project/AllanTools/) for the computation of the ADEV.

Acronyms

ADEV	Allan Deviation
CSV	Comma Separated Values
GHASP	Galileo HAS Parser
GUI	Graphical User Interface
HAS	High Accuracy Service
PPP	Precise Point Positioning

References

European Union (2022) “Galileo High Accuracy Service Signal-In-Space Interface Control Document (HAS SIS ICD)”, Issue 1.0, May 2022, available on-line https://www.gsc-europa.eu/sites/default/files/sites/all/files/Galileo_HAS_SIS_ICD_v1.0.pdf

Fernández-Hernández I., Senni T., Borio D., Vecchione G. (2020) “High-parity Vertical Reed-Solomon Codes for Long GNSS High-accuracy Messages” NAVIGATION: Journal of the Institute of Navigation, June 2020, Vol. 67, N. 2, pp. 365-378, https://doi.org/10.1002/navi.357